컴퓨터학부 20201841 박세연

<creat1>

1. 소스코드

|  |
| --- |
| ssu\_creat\_1.c |
| #include <stdio.h>  #include <stdlib.h>  #include <unistd.h>  #include <fcntl.h>  **int** main(**void**){  **char** \*fname = "ssu\_test.txt"; // 생성하고자 하는 파일명  **int** fd; // 파일 디스트립터  // creat 함수를 이용하여 파일을 생성하고 fd로 파일 디스크립터를 반환 받음  // 이때 0보다 작은 값이 반환되었다면 에러처리 필요  **if** ((fd = creat(fname, 0666)) < 0){  fprintf(stderr, "creat error for %s\n", fname);  exit(1); // 에러 발생 시 강제종료  }  // 성공한 경우  **else**  {  printf("Success!\nFilename : %s\nDescriptor : %d\n", fname, fd);  close(fd); // 파일 닫기  }  exit(0); // 정상종료  } |

1. 실행결과

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

<creat2>

1. 소스코드

|  |
| --- |
| ssu\_creat\_2.c |
| #include <stdio.h>  #include <stdlib.h>  #include <unistd.h>  #include <fcntl.h>  **int** main(**void**){  **char** \*fname = "ssu\_test.txt"; // 생성하고자 하는 파일명  **int** fd; // 파일 디스크립터  // creat 함수를 이용하여 파일을 생성하고 fd로 파일 디스크립터를 반환 받음  // 이때 0보다 작은 값이 반환되었다면 에러처리 필요  **if** (( fd = creat(fname, 0666) ) < 0 ){  fprintf(stderr, "creat error for %s\n", fname);  exit(1); // 에러 발생 시 강제종료  }  // 성공한 경우  // creat로 파일을 생성하면 오직 쓰기(write only)만 가능하기 때문에,  // 이 파일을 다시 읽고 싶으면 파일을 닫았다가 읽기가 가능하게 다시 열어야 함  **else**  {  close(fd); // 파일 닫기  fd = open(fname, O\_RDWR); // open 함수를 이용하여 파일을 읽기가 가능한 플래그로 열어야 함  printf("Succeeded!\n<%s> is new readable and writable\n", fname); // 성공했음을 알리는 출력문  }  exit(0); // 정상종료  } |

2. 실행결과

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

<lseek1>

1. 소스코드

|  |
| --- |
| ssu\_test.txt |
| Linux System Programming!  Unix System Programming!  Linux Mania  Unix Mania |

|  |
| --- |
| ssu\_lseek\_1.c |
| #include <stdio.h>  #include <stdlib.h>  #include <unistd.h>  #include <fcntl.h>  #include <sys/types.h>  **int** main(**void**){  **char** \*fname = "ssu\_test.txt";  off\_t fsize; // 파일 사이즈를 받을 변수 선언  **int** fd; // 파일 디스크립터  // open 함수를 이용해 파일을 읽기 전용으로 열고 fd를 반환받음  // 이때 fd가 음수라면 에러 처리 필요  **if** ((fd = open(fname, O\_RDONLY)) < 0 ) {  fprintf(stderr, "open error for %s\n", fname); // open에서 에러가 발생했음을 알리는 출력문  exit(1); // 강제종료  }  // lseek 함수를 이용하여 파일의 크기를 구할 수 있음  // lseek(fd,0,SEEK\_END) -> 파일 커서가 파일 내용의 가장 끝을 가리키고 있으므로 파일의 크기와 일치한다.  // 이때 fsize가 음수라면 에러 처리 필요  **if** ((fsize = lseek(fd, 0, SEEK\_END)) < 0){  fprintf(stderr, "lseek error\n"); // lseek에서 에러가 발생했음을 알리는 출력문  exit(1); // 강제종료  }    printf("The size of <%s> is %ld bytes.\n", fname, fsize); // 파일명과 파일의 크기를 출력  exit(0); // 정상종료  } |

2. 실행결과

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

<lseek2>

1. 소스코드

|  |
| --- |
| ssu\_lseek\_2.c |
| #include <stdio.h>  #include <stdlib.h>  #include <unistd.h>  #include <fcntl.h>  #include <sys/types.h>  #define CREAT\_MODE (S\_IRUSR | S\_IWUSR | S\_IRGRP | S\_IROTH)  **char** buf1[] = "1234567890";  **char** buf2[] = "ABCDEFGHIJ";  **int** main(**void**){  **char** \*fname = "ssu\_hole.txt"; // 생성하려는 파일명  **int** fd; // 파일 디스크립터  // creat 함수를 이용해 파일을 생성하고 fd로 파일 디스크립터를 반환받음  // 이때 fd가 음수하면 에러 처리 필요  **if** ((fd = creat(fname, CREAT\_MODE)) < 0){  fprintf(stderr, "creat error for %s\n", fname); // creat에서 에러가 발생했음을 알리는 출력문  exit(1); // 강제종료  }  // write 함수를 이용하여 파일에 데이터를 씀  // 오프셋 위치의 기본값은 0이기 때문에 가장 맨 앞을 가리키고 있음  // 따라서 파일의 맨 앞에서부터 12byte만큼의 데이터가 써짐  // write는 기록된 바이트 수를 반환하므로, 정해진 바이트만큼의 데이터가 기록되지 않았다면 에러 처리 필요  **if** (write(fd, buf1, 12) != 12){  fprintf(stderr, "buf1 write error\n"); // buf1을 write하는 데에서 에러가 발생했음을 알리는 출력문  exit(1); // 강제종료  }  // lseek 함수를 이용하여 파일 내의 커서를 이동시킴  // lseek(fd, 15000, SEEK\_SET) -> 현재 가리키고 있는 곳에서부터 15000bytes를 이동시키겠다는 의미  // lseek는 성공 시 위치한 seek pointer 위치를 반환하기에 음수가 반환되면 에러 처리 필요  **if** (lseek(fd, 15000, SEEK\_SET) < 0){  fprintf(stderr, "lseek error\n"); // lseek에서 에러가 발생했음을 알리는 출력문  exit(1); // 강제종료  }  **if** (write(fd, buf2, 12) != 12) {  fprintf(stderr, "buf2 write error\n"); // buf2를 write하는 데에서 에러가 발생했음을 알리는 출력문  exit(1); // 강제종료  }  exit(0); // 정상종료  } |

2. 실행결과

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명